**Chapter 1**

**Introduction**

Let's review each of the four components in-depth to fully appreciate their collective importance of facilitating database interactions.

The create function allows users to create a new record in the database. In the SQL relational database application, the Create function is called INSERT. Remember that a record is a row and that columns are termed attributes. A user can create a new row and populate it with data that corresponds to each attribute, but only an administrator might be able to add new attributes to the table itself.

The read function is similar to a search function. It allows users to search and retrieve specific records in the table and read their values. Users may be able to find desired records using keywords, or by filtering the data based on customized criteria.

The update function is used to modify existing records that exist in the database. To fully change a record, users may have to modify information in multiple fields. In both SQL and Oracle HCM cloud, the update function is simply called "Update".

The delete function allows users to remove records from a database that is no longer needed. Both SQL and Oracle HCM Cloud have a delete function that allows users to delete one or more records from the database. Some relational database applications may permit users to perform either a hard delete or a soft delete. A hard delete permanently removes records from the database, while a soft delete might simply update the status of a row to indicate that it has been deleted while leaving the data present and intact.

**Chapter 2**

**Review of Literature**

The start…

**Chapter 3**

**Requirement Analysis**

**Hardware Requirements**

* Ram 2gb
* Intel Core 2 duo /Greater
* Working Internet
* HDD/SSD:256(minimum)

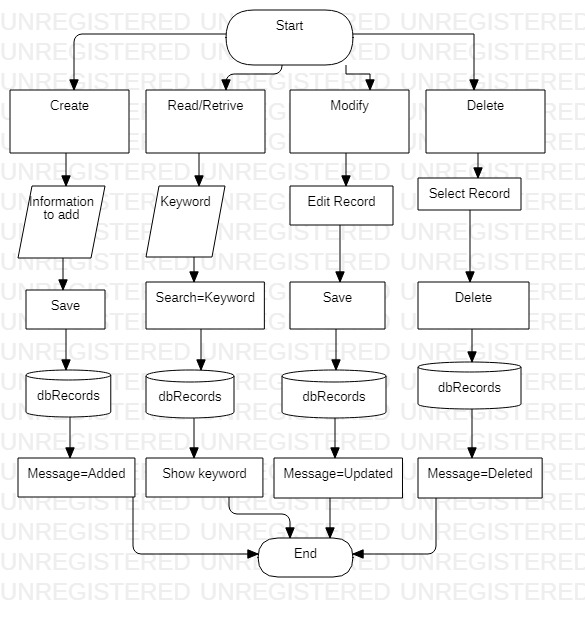
**Software Requirements**

* Latest Version of Chrome / Equivalent
* PostgreSQL
* Python 3.8/ Greater
* TkInter (Python Library)
* Psycopg (Python Library )
* Text Editor (Visual code studio, Sublime etc.)

**Chapter 4**

**Report on Present Investigation**

**4.1 Proposed System**



**4.2Implementation**

**4.2.1 Pseudo code**

from tkinter import Tk, Scrollbar, Button,Label, Listbox,StringVar,Entry ,W,E,N,S, END

from tkinter import ttk

from tkinter import messagebox

import psycopg2 as psy

from db import dbcon

con=psy.connect(\*\*dbcon)

print(con)

cursor=con.cursor()

class todoapp():

def \_\_init\_\_(self):

self.con=psy.connect(\*\*dbcon)

self.cursor=con.cursor()

print(f"You have connected to database")

def \_\_del\_\_(self):

self.con.close()

def view(self):

self.cursor.execute("SELECT \* from todo")

rows=self.cursor.fetchall()

return rows

def insert(self,title):

sql=("INSERT INTO todo(title) VALUES (%s)")

values=[title]

self.cursor.execute(sql,values)

self.con.commit()

messagebox.showinfo(title="TodoList Database", message="New task added")

def update(self,id,title):

tsql='UPDATE todo SET title=%s WHERE id=%s'

self.cursor.execute(tsql,[title,id])

self.con.commit()

messagebox.showinfo(title="TodoList Database", message="Task Updated")

def delete(self,id):

delete ='DELETE FROM todo WHERE id=%s'

self.cursor.execute(delete,[id])

self.con.commit()

messagebox.showinfo(title="TodoList Database", message="Task Deleted")

db= todoapp()

def get\_selected\_row(event):

global selected\_task

index=list\_box.curselection()[0]

selected\_task=list\_box.get(index)

title\_entry.delete(0,'end')

title\_entry.insert('end',selected\_task[1])

def view\_records():

list\_box.delete(0,'end')

for row in db.view():

list\_box.insert('end',row)

def add\_book():

db.insert(title\_text.get())

list\_box.delete(0,'end')

list\_box.insert('end',(title\_text.get()))

title\_entry.delete(0,'end')

con.commit()

view\_records()

clear\_screen()

def delete\_records():

db.delete(selected\_task[0])

con.commit()

view\_records()

clear\_screen()

def clear\_screen():

title\_entry.delete(0,'end')

def update\_records():

db.update(selected\_task[0],title\_text.get())

title\_entry.delete(0,'end')

con.commit()

view\_records()

clear\_screen()

root=Tk()

root.title("CRUD")

root.configure(background="light green")

root.geometry("550x500")

title\_label=ttk.Label(root,text="Task",background="light green", font=("TkDefaultFont",15))

title\_label.grid(row=0,column=0,sticky=W)

title\_text= StringVar()

title\_entry=ttk.Entry(root,width=25,textvariable=title\_text)

title\_entry.grid(row=0,column=1,sticky=W)

add\_btn=Button(root,text="Add Task", bg="blue",fg="white", font="TkDefaultFont 10 bold" ,command=add\_book)

add\_btn.grid(row=0,column=2,sticky=W)

list\_box=Listbox(root,height=16 ,width= 40,font="TkDefaultFont, 13", bg="light blue")

list\_box.grid(row=3,column=1,columnspan=14, sticky= W+E,pady=40,padx=15)

list\_box.bind('<<ListboxSelect>>',get\_selected\_row)

scroll\_bar=Scrollbar(root)

scroll\_bar.grid(row=1,column=14,rowspan=14,sticky=W)

list\_box.configure(yscrollcommand=scroll\_bar.set)

scroll\_bar.configure(command=list\_box.yview)

modify\_btn=Button(root,text="Modify",bg="orange",fg="white",font="TkDefaultFont 10 bold", command=update\_records)

modify\_btn.grid(row=15,column=1)

delete\_btn=Button(root,text="Delete",bg="purple",fg="white",font="TkDefaultFont 10 bold", command=delete\_records)

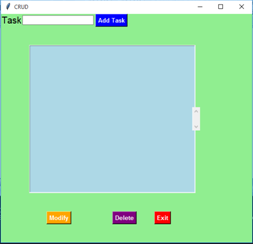
delete\_btn.grid(row=15,column=2, padx=35)

exit\_btn=Button(root,text="Exit",bg="red",fg="white",font="TkDefaultFont 10 bold", command=root.destroy)

exit\_btn.grid(row=15,column=3)

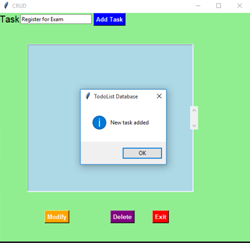
view\_records()root.mainloop()

**4.2.2 Screenshots of the output :**

****

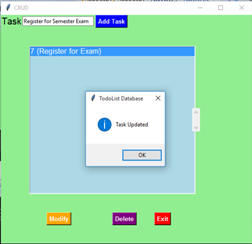
**4.2.2.1 Output**

**Fig 4.2.2.1** Shows the Format and Representation of our application

****

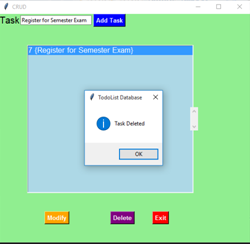
**4.2.2.2 Output of Add event**

**Fig 4.2.2.2** In this fig it represents the outcome of the event of adding the task and a dialogue box conforming the successful execution of the event

****

**4.2.2.3 Output of Modify event**

**Fig 4.2.2.3** In this fig it represents the outcome of the event of updating the task and a dialogue box conforming the successful execution of the event

****

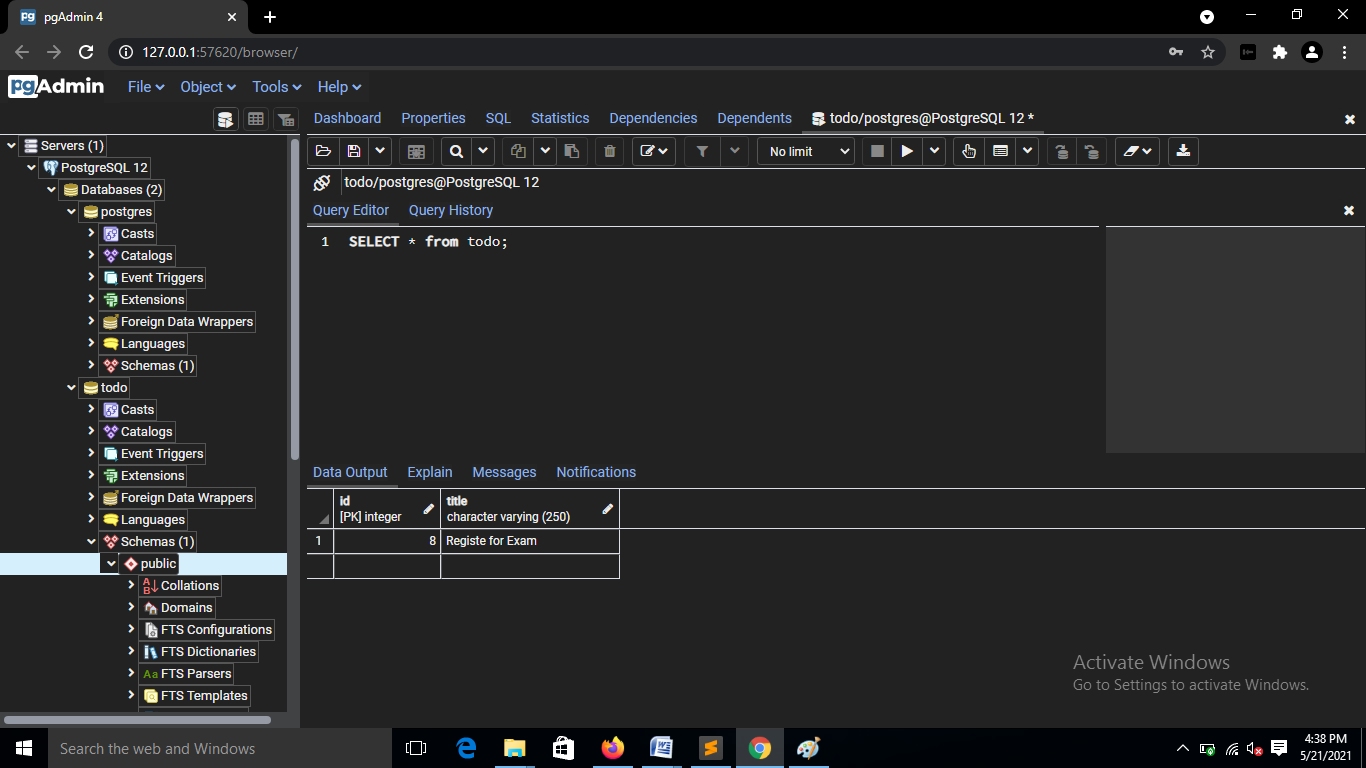
**4.2.2.4 Output of Delete event**

**Fig 4.2.2.2** In this fig it represents the outcome of the event of deleting the task and a dialogue box conforming the successful execution of the event

**Chapter 5**

**Results and Discussion**

**Results:**

****

**5.1 Server Database**

As a result we can see that the CRUD application as been developed successfully with the help of Python 3.9 and Python libraries such as TkInter for the GUI and end result and Psycopg for the database connectivity with the PostgreSQL

**Fig 5.1** Shows the database connection was established successfully and the results can been seen on PostgreSql database

**Chapter 6**

**Conclusion**

Between the document oriented databases PostgreSQL has an overall very good performance time for the INSERT, UPDATE and DELETE operations, but it falls behind when it comes to modelling the data. This database does not support nested documents. For applications doing intensive write operations a good choice would be PostgreSql.

Database management systems are applications that could change, in time, the way they handle the CRUD operations and this could enhance or degrade their performance. When developing applications it is recommended to make the choice that best fits the requirements after a careful analysis of all features it offers.

CRUD testing is a black box testing technique to validate the functionality of a software product. This term for database testing is applicable for SQL and other databases, and ensures proper data mapping, data integrity, maintaining of ACID properties, and accuracy of business rules.

**References**

[1] Ciprian-Octavian Truic , Florin R ̆adulescu, Alexandru Boicea and Ion Bucur, Performance evaluation for CRUD operations in asynchronously replicated document oriented database, Department of Computer Science, Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Bucharest, Romania

# [2] Kunal Relan, CRUD Application with Flask, Apress, Berkeley, CA

# [3] Sanjib Sinha, A CRUD Application, Apress, Berkeley, CA

[4] Velte, Toby; Velte, Anthony; Elsenpeter, Robert C., Cloud Computing, A Practical Approach